INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

CONTROL OF A MOBILE PLATFORM DIDACTIC PURPOSES

Student:

Seif Eddine Lazghab - a46656

Master Degree In Industrial Engineering

Supervised by:

Prof. José Gonçalves
Prof. José Lima

Bragança 2021

CONTROL OF A MOBILE PLATFORM DIDACTIC PURPOSES

Seifeddine Lazghab

Dissertation presented to the Escola Superior de Tecnologia e Gestão of the Instituto Politécnico de Bragança, to obtain the Master's Degree in Industrial Engineering

Bragança 2021

Dedication

To my dear parents, for their love, sacrifices and support in the most difficult moments, which are at the origin of our success, may god keep and protect them.
To our dear brothers, for their constant encouragement and all the help they give us on a daily basis.
To all the people who, from near or far, have participated in our work.To the whole Electrical and Industrial family,

Our paths crossed for the first time when we entered the IPB , keeping the friendship and international relation that unites us and the memories of all the moments we spent together, nothing to say.

*W*e dedicate this modest work to them

Acknowledgements

It is with great pleasure that I would like to thank all those who, in one way or another, contributed to the realisation of this final work in IPB.

My sincerest thanks to Dr ***José Gonçalves*** *and Dr* ***José Lima*** *my university supervisors, for the enriching and interesting experience they gave me during the period of the project and for the autonomy they gave me throughout the project.*

I would like to thank all those who helped me in the elaboration of this work, my friends,and my family.

Finally, I would like to thank the entire teaching team of the IPB for having ensured a good coaching and education during one year of research master.

Abstract

Robots are electromechanical machines having ability to perform tasks or actions on some given electronic programming. Line follower robots are mobile robots having ability to follow a line very accurately having an onboard hardwired control circuit. while Omni directional mobile robots have been popularly employed in several applications.This situation brings the idea of omnidirectional robot at manufacturing. Such a robot can respond more quickly and it would be capable of more sophisticated behaviors such as to transport materials and placed on processing machine and outgoing warehouses. This thesis has tried to focus in the control of four wheel omnidirectional mobile robot to be applied to the Factory Lite competition. Four motors are used for governing wheel's motion. Practical applications of a line follower and odometry will be implemented in this work.

Keywords:

Omni Robot, Encoders ,Odometry ,Omni wheels ,Velocity control , PID control

Résumé

Les robots sont des machines électromécaniques capables d'exécuter des tâches ou des actions selon une programmation électronique donnée. Les robots suiveurs de ligne sont des robots mobiles capables de suivre une ligne avec une grande précision grâce à un circuit de contrôle câblé embarqué. Les robots mobiles omnidirectionnels sont couramment utilisés dans plusieurs applications, ce qui amène l'idée d'un robot omnidirectionnel dans la fabrication. Un tel robot peut Un tel robot peut répondre plus rapidement et il serait capable de comportements plus sophistiqués tels que transporter des matériaux et les placer sur des machines de traitement et des entrepôts de sortie. Cette thèse a essayé de se concentrer sur le contrôle d'un robot mobile omnidirectionnel à quatre roues pour être appliqué à la compétition Factory Lite. Quatre moteurs sont utilisés pour gouverner le mouvement des roues. Des applications pratiques d'un suiveur de ligne et d'odométrie seront mises en œuvre dans ce travail.

Mots-clés :

Robot Omni, encodeurs, odométrie, roues Omni, contrôle de vitesse, contrôle PID.

Contents

List of Tables

List of Figures

Acronyms

AGV Automatic Guided Vehicles. 12

AI Artificial Intelligence. 8

AIV Autonomous Intelligent Vehicle. 12

EOAT End Effector Arm Tooling. 11

FMS Flexible Manufacturing Systems. 11

ICSP In Circuit Serial Programming. 26

IDE Integrated Development Environment. 27

IPB Instituto Politécnico de Bragança. 1

LED Light Eitting Diode. 31

LPS Local Positioning System. 15

MOSFET Metal Oxide Semiconductor Field Effect Transistor. 31

PID Proportional Integral Derivative. 39

PWM Pulse Width Modulation. 26

RFID Radio Frequency Identification. 18

TCP/IP Transmission Control Protocol/Internet Protocol. 32

TTL Transistor-Transistor Logic. 29

UART Universal Asynchronous Receiver-Transmitter. 26

USB Universal Serial Bus. 26

Chapter 1

Introduction

In this work, we will study and elaborate an autonomous mobile robot. this work will start with a general study on mobile robots after the phase of the electrical and mechanical design study and design and conclude with the choice of material the electrical and data-processing part. the work will be elaborated within the laboratory of the Instituto Politécnico de Bragança (IPB) .

The main objective of this proposal is to control a four omnidirectional wheel mobile robot. It should be controlled automatically during the development stage. The platform should be able to support further developments.

Mobile robotics has known a great revolution. It has become indispensable and these applications have spread to almost all areas of life. Mobile robotics will of course continue to integrate the progress of computer science, and will first benefit from increasingly complex algorithms that can be executed in real time. But also, via mobile Internet access techniques. Mobile robots will also benefit from the development of mechanics and electronics. It will be composed of intelligent sensors, micro controllers and motors with encoders.

Robotics allows individuals to be assisted with tough or repetitive tasks. Furthermore, it is a dream to be able to replace the operator with a machine in these jobs. Robots observation and thinking abilities are improving every day, and they will be relied upon to play an increasingly vital part in human lives in the future. And being able to carry out

the functions necessary inside the university (delivery of objects, monitoring). We will investigate thoroughly and decide the good mechanical components, taking into account the complexity of the ground.

Omnidirectional mobile robots are becoming increasingly popular in mobile robot applications, since they have some distinguishing advantages over nonholonomic mobile robots. They have simultaneously and independently controlled rotational and translational motion capabilities, which means that they can move at each instant in any direction without reorientation [1].

The perception and reasoning faculties of robots are progressing every day now and even more so in the future, they are called upon to play an increasingly important role in our lives. Within the framework of our project it is a question of realizing a four omnidirectional mobile robot, and to be able to carry out the tasks required during the factory lite competition. Considering the difficulty of the ground which is full of laying we will try well and determine the good program to pick and place the materials and parts in the area of the competition.

Chapter 2

State Of The Art

The context of the project, as well as the goal, are presented in this chapter. The framework of the project will then be presented, along with a study of the history of robotics and the various fields of application of robots in order to determine the problem on which this thesis is based.

2.1 History of Robotics

Without going back to the first concepts of machines replacing man as early as the 17th century, robotics was born, in the 1950s, from the intersection of the needs and availability of new technologies developed during the Second World War: electronics, automation and computing. The first two orientations of these machines were to meet the needs of manufacturing industry and the needs of industry in environments hostile to man.

Mobile robotics is a new field. Mobile robots range from the sophisticated space robots, to the military flying robots, to the lawn mower robots at our backyard. Mobile robotics is based on many engineering and science disciplines, from mechanical, electrical and electronics engineering to computer, cognitive and social sciences. A mobile robot is an autonomous or remotely operated programmable mobile machine that is capable of moving in a specific environment. Mobile robots use sensors to perceive their environment and make decisions based on the information gained from the sensors. The autonomous

nature of mobile robots is giving them an important part in our society. Mobile robots are everywhere, from military application to domestic applications. The first mobile robots as we know them today were developed during World War II by the Germans, and they were the V1 and V2 flying bombs. In the 1950s, W.Grey Walter developed Elmer and Elsie, two autonomous robots that were designed to explore their environment. Elmer and Elsie were able to move towards the light using light sensors, thus avoiding obstacles on their way. The evolution of mobile robots continued and in the 1970s Johns Hopkins University develops the "Beast". The beast used an ultrasound sensor to move around. During the same period, the Stanford cart line follower was developed by Stanford University. It was a mobile robot that was able to follow a white line, using a simple vision system. The processing was done off-board by a large mainframe. The most known mobile robot of the time was developed by the Stanford Research Institute, and it was called Shakey. It was the first mobile robot to be controlled by vision. It was able to recognize an object using vision, find its way to the object. These robots had limitations due to the lack of processing power and the size of computers, and thus industrial robotics was still dominating the market research. Industrial manipulators are attached to an off-board computer for their processing requirements and thus do not require an on-board computer for processing. Unlike industrial robots, mobile robots operate in dynamic and unknown environment and thus require many sensors. And therefore more processing power. Another important requirement of mobile robots is that their processing must be done on board the moving robot and cannot be done off-board. The computer technology of the time was too bulky and too slow to meet the requirements of mobile robots. Also, sensor technology had to advance further before it could be used reliably on mobile robots. In the last twenty years we saw a revolution in computer technology. Computers got smaller, a lot faster and less expensive. This met the requirements of mobile robots and as a result we saw an explosion of research and development activities in mobile robotics. Mobile robots are increasingly becoming important in advanced applications for the home, military, industry, space, and many others. The mobile robot industry has grown enormously, and it is developing mobile robots for all imaginable applications. The vast number of mobile

robot applications has forced a natural subdivision of the field based on their working environment : land or surface robots,aquatic/underwater robots, aerial robots and space robots. Land and surface robots are subdivided based on their locomotion: Legged robots, wheeled robots and track robots. Legged robots can be classified as two legged robots and animal-like robots that can have anywhere from four legs to as many as the application and the imagination of the developer requires [2].

2.2 Evolution of Mobile Robotics

The revolution of mobile robotics has increased the need for more mobile robotics engineers for manufacturing, research, development and education. And this in turn has significantly changed the nature of engineering and science education at all levels, from K-12 to graduate school. Mobile robotics are widely accepted as a multidisciplinary approach to combine and create knowledge in various fields as mechanical engineering, electrical engineering, control, computer science, communications, and even psychology or biology in some cases. The majority of robotics research is focusing on mobile robotics from surface robots, humanoids, aerial robots, underwater robots, and many more. The development of several less expensive mobile robotic platforms [2].

2.3 Robotics Application

In order to better define the scope of personal and service robotics, as considered in this study, it seems important to illustrate it with an example.

2.3.1 The robots of intervention

Intervention robots are generally remotely operated (remotely operated) by direct commands (joysticks, master arm and other physical or virtual control devices), or in semi-autonomy by high-level commands to perform and sequence tasks. They are used to perform tasks in environments that are difficult to access or dangerous for humans.

- Defense: theater of operations with ground robots and aerial drones. Most military robots fall within the field of intervention robots. They carry out reconnaissance, surveillance, mine clearance or destruction functions.

- Civil security: Robotics for civil security is used in particular during interventions on natural disasters. In particular, exploration robots are used to explore inaccessible places. There are also robots used by law enforcement agencies for defusing or destroying parcel bombs.

- Nuclear: It mainly concerns intervention in irradiated environments, harmful to the human operator. It has given rise to the development of "hardened" technologies to resist higher or lower levels of radiation.

- Submarine: Underwater robotics is also an important field of development for military (underwater surveillance), petroleum (exploration, exploitation) applications. Underwater robots, as for space exploration, have the advantage of do not require the transport of human operators.

- Inspection and maintenance : in specific environments (pipeline, etc.). Inspection in specific environments is concerned with intervention in places where humans cannot materially intervene at an acceptable cost. In particular, the repair of leaks in pipes is a relatively common application of robotics, which avoids civil works for human intervention.

- Space exploration : Robots are today the preferred means of space exploration. Missions to Mars are a significant example of this: information transmission delays do not allow real-time remote operation by a remote manipulator (on Earth). This type of application is the bearer of significant innovations in terms of perception and robustness.

2.3.2 Professional service robotics

Professional service robotics intervenes in assistance to the worker in a professional setting. Its functions are mainly to relieve professionals of repetitive or dangerous tasks (in a perspective close to industrial robotics), or to assist them in interventions which require a level of precision or qualities inaccessible to the human operator.

- Cleaning robot : These robots are an important part of professional service robotics. These robots are used in particular in public or domestic spaces
- Construction and demolition : Construction robots are used more and more for specific operations in building and civil engineering. They are generally associated with a particular construction technique (concrete spraying for example). In all cases, these are special machines developed for a particular application.
- Logistics robot : Logistics robots are also an interesting development path, in particular because of the possibility of developing machines produced in series: logistics is a vast sector, which implements partly standardized procedures and equipment.
- Public relations robot : Reception or assistance robots in public places are today the subject of experimentation rather than real commercial development at the level of a sector. Public relations robots are implemented in places such as museums or shopping centers, to help visitors find their way around, to provide them with information.
- Medical robot: The medical field is also an important market for the development of professional service robotics. This is a market open to innovations, in high demand and with specific expectations from healthcare professionals and patients. The areas of development aim to assist doctors (surgical assistance robots for example), paramedical staff (assistance in handling bedridden people, robotic wheelchairs), patients (rehabilitation assistance, robotic prostheses or orthotics - including exoskeletons in the long term).

2.4 Robots Classification

During the course of history we can distinguish 3 types of robots corresponding in some way to the evolution of this "species" created by Man. The first type of machine that we can call robot corresponds to the "Automata". These are generally programmed in advance and allow performing repetitive actions [1].

The second type of robot corresponds to those equipped with sensors. There are temperature, photo-electronic, ultrasonic sensors for example to avoid obstacles and/or to follow a path. These sensors will allow the robot a relative adaptation to its environment in order to take into account random parameters that could not have been considered during their initial programming. These robots are therefore much more autonomous than PLCs, but require a more significant investment in design time and money.

Finally, the last type of existing robot corresponds to those with so-called Artificial Intelligence (AI) based on complex mathematical models such as neural networks. In addition to physical sensors like their predecessors, these robots can make much more complex decisions and also rely on learning from their errors as humans can do. Of course, it will be a long time before the most "intelligent" robot will be equal to humans in its adaptability and decision making.

2.5 Mobile Robot

A mobile robot is a mechanical, electronic and computer system that physically acts on its environment to achieve an assigned objective. This machine is versatile and capable of adapting to certain variations in its operating conditions. It is equipped with functions of perception, decision and action. Thus, the robot should be able to perform various tasks in different ways and perform its task correctly, even if it encounters new and unexpected situations.

The name Mobile Robot includes all types of robots that have the ability to move, which is the common characteristic between them, the difference lies in the way, which

depends on the area of use of the robot, by which the robot will achieve this ability of movement. Mobility by wheels is the most commonly applied mechanical structure. Depending on the arrangement and dimensions of the wheels, this technique ensures movement in all directions with high acceleration and speed.

Figure 2.1: Kuka mobile robot[3]

2.5.1 Components of a mobile robot

The Motors

A motor makes it possible to initially carry out a rotational movement. This movement is communicated by means of the motor shaft and can then be transformed into a translational movement by means of various technological solutions such as screw/nut, pinion/rack, connecting rod/crank. The choice of one motor or another depends on the requirements. It is necessary to know the power absorbed under load, the mechanical torque and the rated speed to select one motor over another.

The Control electronics

The realization of electronic boards is already a difficulty in itself, but in the case of robotics, it is coupled with a reduced space requirement, because their size must be minimized as much as possible. As the electronics are a very sensitive part of the robot, they must be protected. All connections between the boards, sensors, motors and power supply are sensitive points.

Sensors

Sensors are the sensory organs of a robot. Some are fragile and must be protected, others on the contrary must be able to absorb shocks. The simplest ones can be directly connected to the control center, like switches. The other types require a small adapter interface, such as infrared or ultrasonic sensors. Other more sophisticated types require a special card, such as cameras. There are two types of sensors:

- The external sensors: These are exteroceptive sensors, delivering information related to the environment or to the interactions between the robot and its environment, such as distance sensors.
- The internal sensors : These are the sensors that provide information on the internal state of the robot: wheel position or speed sensors and battery charge sensors.

The Power interfaces of the motors

Motors can only be controlled directly through a power interface. Knowing its characteristics allows us to choose the best interface. A particularly important point is the heat dissipation of the transistors or integrated circuit.

Control centers

The control center is generally a board equipped only with a processor or microcontroller. In this case, the control center consists of the processor or microcontroller and its memory devices. But sometimes it is completed by interfaces for motors and sensors.

2.6 Industrial Robot

Because they can be programmed to perform dangerous, dirty and/or repetitive tasks with consistent precision and accuracy, industrial robots are increasingly used in a variety of industries and applications. They come in a wide range of models with the reach distance, payload capacity and the number of axes of travel (up to six) of their jointed arm being the most common distinguishing characteristics.

In both production and handling applications, a robot utilizes an End Effector Arm Tooling (EOAT) attachment to hold and manipulate either the tool performing the process, or the piece upon which a process is being performed.

The robot's actions are directed by a combination of programming software and controls [4].

Figure 2.2: Industrial robots

2.7 Autonomous Vehicle Guided

Manufacturing companies are constantly striving towards more efficient and cheaper ways to produce their products. Flexible Manufacturing Systems (FMS) are employed in the pursuit of a more cost effective and time efficient process [1] . The setting up of a new production line represents a major investment for a company, and to make minor changes

Figure 2.3: Autonomous vehicle guided

to existing production lines can result in down time, reduced productivity and require a large investment of capital.

The Automatic Guided Vehicles (AGV) is the major component of this flexible production line. The term Automatic Guided Vehicles refers to vehicles that are able to navigate without human intervention [4] [5] [6] . It could be used as an alternative to the fixed conveyor belt which passes an unfinished product through the production stages sequentially. The AGV could pick and place an unfinished product to a different production stage allowing for a more flexible production line. The latest version of an AGV is often referred to as an Autonomous Intelligent Vehicle (AIV). Figure 2 outlines a production line with an AIV incorporated [5].

2.8 Collaborative Robot

Collaborative robotics has shown great promise to bring potentially complex tasks in frequently changing settings closer to automation. In that respect, especially tasks involving contact between the robot and the environment such as assembly have remained challenging. This is due to the fact that contact states are hard to detect and due to the difficulty in modeling the effects of the robots' actions. Reinforcement learning of local control policies

has proven to be a promising method to obtain control policies for interaction tasks. Of particular importance here is the sample efficiency as explorative actions are costly and potentially hazardous. An open issue remains the generalization of learned policies to novel settings. We see the potential of addressing this using a-priori (partial) knowledge of the robots' model perform learning in task-invariant operational spaces. Recent policy learning approaches also achieved a tight coupling with perception [Levine et al., 2016]. From the perception point of view, effective and flexible use of multi-sensory data in real-time will be necessary. Although sensor fusion has been demonstrated in other areas (mapping and localization), physical interaction suffers from the challenges outlined in the previous paragraph and many of the existing methodologies for sensor fusion do not meet all the challenges that physical contact, including both rigid and deformable objects, brings [6].

2.9 Omnidirectional Robot

Omni-directional mobile robot is a kind of holonomic robot. Compared with more common car like (nonholonomical) mobile robot, omni-directional mobile robot has the ability to move simultaneously and independently in translation and rotation [5]. The maneuverability of the omni-directional mobile robot makes it widely studied in the dynamic environmental applications. The annual international Robocup competition in which the team of autonomous robots compete in a soccer like game, is an example where the omni-directional mobile robot can be used. The Ohio University (OU) Robocup Team's entry Robocat is a cross-disciplinary research project (including ME, EE, CS students and faculties) for Robocup small-size league competition. The current OU Robocup team members are Phase V omnidirectional mobile robot, as shown in Figure 1-1. The Phase V Robocat is an omini-directional robot with three orthogonal wheels, arranged 120' apart. Each wheel is driven by a DC motors installed with shaft optical encoder. The robot is operated by an PC104 computer with 486 processors running Linux operating system. A roof camera over the play field can sense the position and the azimuth angle of robots. From the robot testing

and competition at the Robocup games, it is realized that a precise trajectory control for the robot is one of the key areas to improve the team's performance. The trajectory control of the omni-directional mobile robot can be divided into two tasks, trajectory planning and trajectory following. Trajectory planning is to build a feasible and optimal geometric path. Trajectory following is to use feedback [7]

Figure 2.4: Omni robot

2.10 Odometry In Relation To Mobile Robotics System

A mobile robot is an automatic machine that accomplishes a task in a given environment and recognizes its surroundings with multiple sensors. Positioning of the mobile robot to accomplish a given task and achieve autonomous travel is an important technique and currently an important research field. There are two generals methods used for positioning [1]: relative positioning and absolute positioning. Relative positioning, also known as dead reckoning, calculates the position and heading angle using odometry or inertial sensors. Absolute positioning calculates the position using an external distance measuring system. Dead reckoning calculates the relative position from the initial starting point information. The encoders attached to the wheels of the robot measure its angular rate. The position and the heading angle of the mobile robot can be calculated by this angular rate. This method is known as odometry and is the most widely used method for the positioning [2-16]. Odometry calculates the position and the heading angle by the integration of the travel distance and heading angle rate and has systematic and

nonsystematic errors [2,3]. Therefore, errors in the position and heading angle increase continuously as the operating time and moving distance increase because these errors are accumulated by integration. Currently, various researchers have worked to reduce these errors by modeling and filter design [4-7], calibration algorithms [8-11], and various combined sensor systems [12-16]. The aim of these studies was to extend the period of navigation, without the help of external absolute position information. However, the best solution to overcome the accumulated positioning error is to periodically compensate with the external absolute position information. A Local Positioning System (LPS) for a mobile robot based on ultrasonic transmission is widely used for robotics applications because it is simple, inexpensive, and provides relatively accurate position. Various methods exist for positioning based on ultrasonic transmission [8].

2.11 Conclusion

After presented the mobile robotics history in the world, we described briefly the evolution of robotics and the techniques used. In the following chapter we will present the framework of the competition that we will follow for the project.

Chapter 3

Project Framework

In this chapter we will present the general framework of the robotic competition, starting with the game area, the components of the area, and we conclude with a game strategy in order to translate it into an Arduino program at the end of this project.

3.1 The Competition

The Robot@factory robotics competition, which was included in Robotica (the main Portuguese robotics competition). The robot competition takes place in an emulated factory, where automatically guided vehicles must cooperate to perform tasks. To accomplish their goals the AGVs must deal with localization, navigation, scheduling and cooperation problems that must be solved autonomously. The presented robot competition can play an important role in education due to the inherent multi-disciplinary concepts that are involved, motivating students to technological areas. It also plays an important role in research and development, because it is expected that the outcomes that will emerge here, will later be transferred to other application areas, such as service robots and manufacturing [9] .

The competition is divided into three rounds, preferably held on consecutive days. Each team will have 10 minutes to do the initial tests on the field before the trial starts. During the trial a team can attempt as much runs as it is possible in its 10 minutes

slot. For each trial, the final score is the total number of parts correctly placed in the outgoing warehouse. The best run is automatically considered. The time to finish plus any additional time penalization is used as the next criteria. The figure 9.1 shows the starting area for the robot and the machines types with the input and output places. For each run, the robot must start inside the green area [10] .

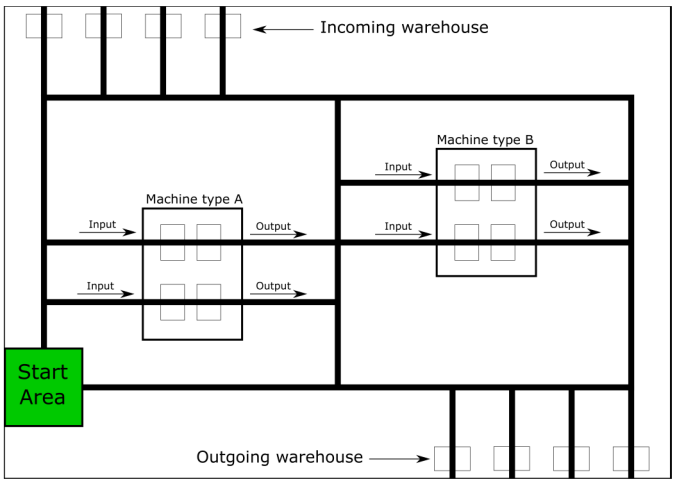

Figure 3.1: Area of the competition

3.2 The Machines and the Warehouses

On each machine there is an area where the parts should be placed to be processed (Input) and another one where the processed parts should be picked (Output) as illustrated in figure 4.1. It is the robot's responsibility of the loading and unloading of the parts into the machines. After the part is placed on the left side of the machine (Input), it will be processed and should be picked on the right side (Output) [10].

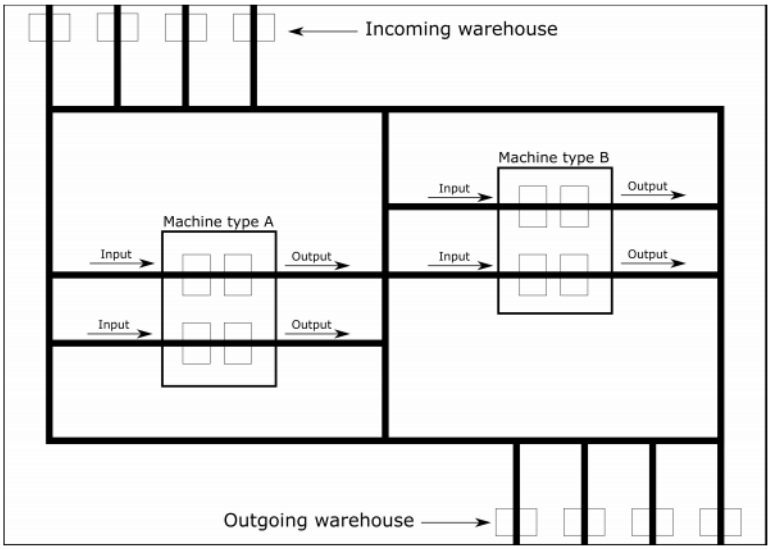

Figure 3.2: The machines and wearhouses

3.3 First Round

In the first round, the objective is just to collect the four parts from the incoming warehouse and transport them to the outgoing warehouse as fast as possible. The four parts will be already placed on the incoming warehouse, ready to be moved.

3.4 Second Round

In the second round, some of the four parts present in the incoming warehouse must be placed in a machine for processing. After the completion of this operation they can be carried into the outgoing warehouse. A table that maps the Radio Frequency Identification (RFID) codes that differentiates the parts from those that are already processed and can be taken directly to the outgoing warehouse will be published. For this round, there is an additional rule that every attempt must be spaced by at least one minute. Is only possible to resume an attempt one minute after the moment when the previous attempt was initiated. This limits the maximum number of attempts to less than 10 [10].

3.5 Line Detection

A line is often defined as a line or even a curve, which may be more accurate. Because this definition boils down to saying that a line is a series of dots. True, but in reality, a line is observed when the line is thick enough and large enough for our QTR sensor can see it. Moreover, it is imperative that it has a color that is quite different from the background color. In our case, the line is black. The thickness of the line is sufficient for the robot's sensor to detect it but moderately so that it is not considered as a full figure.

3.5.1 Line detection principle

We know that radiation of a certain wavelength will be absorbed by a material, if the material diffuses all radiation except that of this wavelength, and it will be diffused by another material diffusing radiation, one of which has a wavelength of. In our case, we used two colors: black and white. As for the QTR sensor of our robot, they will only have to be able to detect the absence or the presence of the line. The QTR sensor will detect the presence or absence of a light beam. It is necessary to take in consideration the environment of our robot, which is not necessarily in total darkness; the only significant sources of light in the way of the competition come from neon lights or lamps.

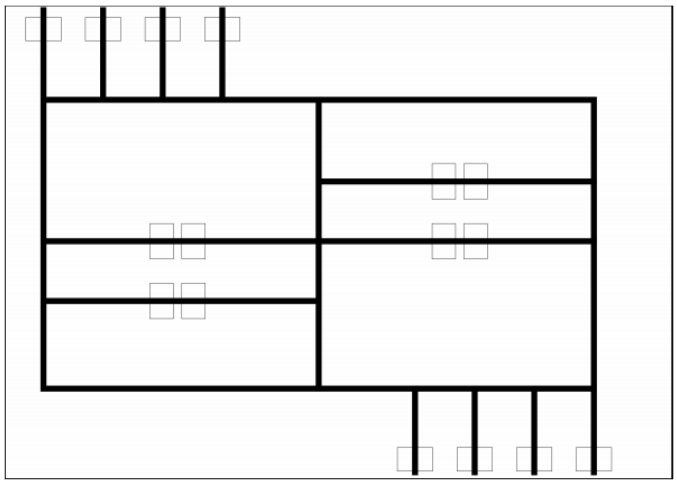

Figure 3.3: Lines to follow

3.5.2 Line following

In this first case, the robot knows that it is in the presence of a line. Moderately curved. Then checks which of the lateral sensors detects the line.

1. If it is the straight sensor that detects the line, it means that it is deflecting through the left, so it must rotate to the right.

2. If the left sensor detects the line, it means the opposite: it deviates by the right and must therefore rotate to the left.

This rotation continues until the central sensors has detected the line. And if it detects it, the rotation stops and the adjustment to the line is completed. Program ending then it switches back to the line processing program so that the robot continues to follow the rest of the line.

3.6 Dynamic Modelling

3.6.1 Kinematic model

Most of kinematic models of mobile robots assume that no tire slippage occurs, so the inputs to the system are right and left wheel angular velocities, Wr and Wl , respectively. Then the motion of the robot can be described by the simple kinematics of rigid bodies. In order to determine the robot motion, it is so important to define the position and orientation of the robot as the location and orientation of the center of gravity.

3.7 The Killough Drive

A symmetrical holonomic drive with four omni wheels is usually called a 'Killough Drive'. This is shown in an 'X' configuration below, but it works in a similar way if it's rotated by 45° into a '+' configuration, where the motor angles would be 0°, 90°, 180° and 270°. One thing to bear in mind with more than three wheels is that some form of chassis flexibility

or suspension will help to keep all the wheels touching the ground in cases where the floor isn't mirror-flat. Omni-wheel robots though, are probably somewhat better-behaved than Mecanum wheel robots when a wheel does lose contact with the ground.

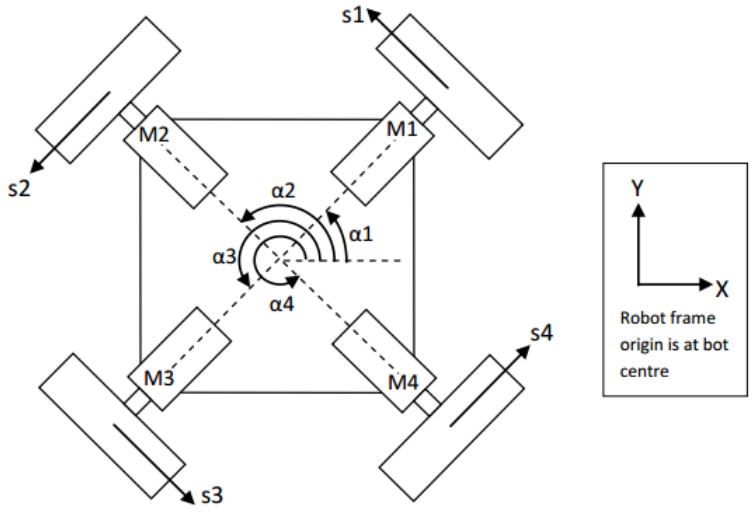

Figure 3.4: Omni wheel x structure

In the 'X' configuration, the angle of each motor axis from the robot coordinate frame 'x' axis is:

$$\alpha 1 = 45^\circ \ \alpha 2 = 135^\circ \ \alpha 3 = 225^\circ \ \alpha 4 = 315^\circ$$

We add $\pi/2$ to get the drive direction of each wheel:

$$w1 = \alpha 1 + \pi/2 = 135^\circ$$

$$w2 = \alpha 2 + \pi/2 = 225^\circ$$

$$w3 = \alpha 3 + \pi/2 = 315^\circ$$

$$w4 = \alpha 4 + \pi/2 = 45^\circ$$

Once using the trig to express the direction of each wheel as components in x and y

relative to the robot coordinate frame:

For wheel 1,

$$x1 = cos(\alpha 1 + \pi/2) \cdot s1$$

and,

$$y1 = sin(\alpha 1 + \pi/2) \cdot s1$$

and the same for the other three wheels. Then gather up all the individual motor contributions to the robot motion in x, in y and in w:

$$x = x1 + x2 + x3 + x4;$$

$$y = y1 + y2 + y3 + y4;$$

$$w = s1 + s2 + s3 + s4$$

To use the summed x and y contributions we need to substitute in the trig expressions we worked out that express the x and y contributions of each motor in terms of $'\alpha'$ and 's' to get:

$$x = cos(\alpha 1 + \pi/2) \cdot s1 + cos(\alpha 2 + \pi/2) \cdot s2 + cos(\alpha 3 + \pi/2) \cdot s3 + cos(\alpha 4 + \pi/2) \cdot s4 \quad (3.1)$$

$$y = sin(\alpha 1 + \pi/2) \cdot s1 + sin(\alpha 2 + \pi/2) \cdot s2 + sin(\alpha 3 + \pi/2) \cdot s3 + sin(\alpha 4 + \pi/2) \cdot s4 \quad (3.2)$$

3.7.1 kinematic Model

In order to find motion models for a surface vehicle, the pose of the vehicle must be identified as (x, y, θ) and associated velocities are

$$v_x(t) = \frac{dx_t}{d(t)} \ ; \ v_y(t) = \frac{dy_t}{dt} \ ; \ w(t) = \frac{d\theta_t}{dt} \quad (3.3)$$

$$\begin{bmatrix} v(t) \\ vn(t) \\ w(t) \end{bmatrix} = \begin{bmatrix} cos(t) & sin(\theta(t)) & 0 \\ -sin(\theta(t)) & cos(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} v(t) \\ vn(t) \\ w(t) \end{bmatrix} \tag{3.4}$$

3.7.2 Relationship between wheel velocity

The relationship between the wheels velocities v0, v1, v2 and v3, with the robot velocities v, vn and w is described by the following equation:

$$\begin{bmatrix} v_0(t) \\ v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & d \\ -1 & 0 & d \\ 0 & -1 & d \\ 1 & 0 & d \end{bmatrix} \times \begin{bmatrix} v(t) \\ vn(t) \\ w(t) \end{bmatrix} \tag{3.5}$$

It is possible to obtain the equations that determine the robot velocities related with wheels velocity but the matrix associated with equation (2.3) is not square. This is because the system is redundant [9]. It can be found that:

$$\begin{aligned} v(t) &= (\tfrac{1}{2}) \cdot (v_3(t) - v_1(t)) \\ vn(t) &= (\tfrac{1}{2}) \cdot (v_0(t) - v_2(t)) \\ w(t) &= (\tfrac{1}{2 \cdot d}) \cdot (v_0(t) + v_1(t) + v_2(t) + v_3(t)) \end{aligned} \tag{3.6}$$

3.8 Architecture Diagram

The electrical part in relation to the mechanical and control part is detailed in the figure 3.5. The electrical configuration is quite simple to set according to the program loaded into the microcontroller.

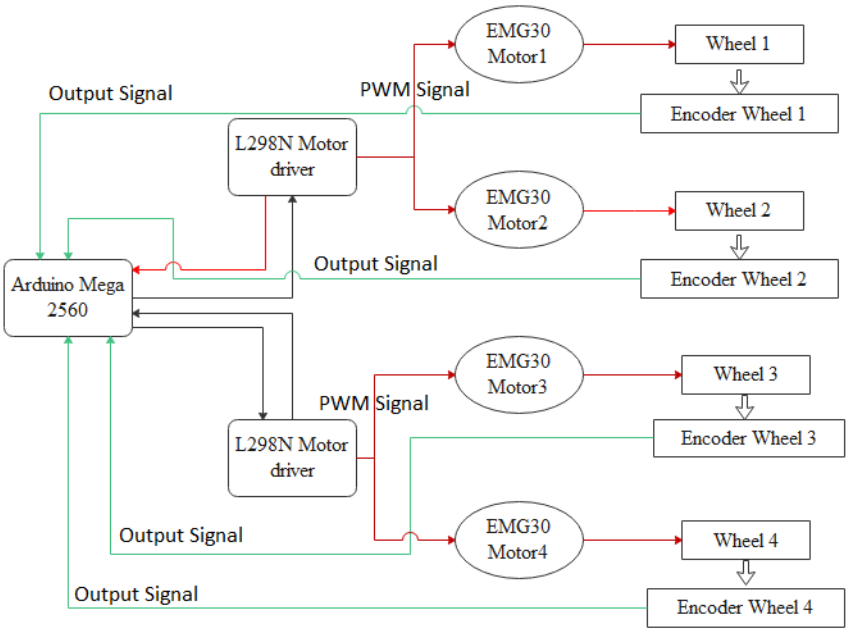

Figure 3.5: Bloc Presentation of the robot

3.9 Programing with Interruption

Interrupts are useful for making things happen automatically in micro-controller programs and can help solve timing problems. Good tasks for using an interrupt may include reading a rotary encoder, or monitoring user input.

To ensure that a program always caught the pulses from a rotary encoder, so that it never misses a pulse, it would make it very tricky to write a program to do anything else, because the program would need to constantly poll the sensor lines for the encoder, in order to catch pulses when they occurred. Other sensors have a similar interface dynamic too, such as trying to read a sound sensor that is trying to catch a click, or an infrared slot sensor (photo-interrupter) trying to catch a coin drop. In all of these situations, using an interrupt can free the micro-controller to get some other work done while not missing the input [11].

3.9.1 Type of interruption

Interrupts really enhance the use of microcontrollers in a big way. Interrupts make the programs react to the hardware of the microcontrollers, which may be a reaction from the circuit/environment outside of the microcontroller. An interrupt is a condition that causes the microprocessor to temporarily work on a different task, and then later return to its previous task.

Internal interruption

An internal interrupt is a type of interrupt that results from a specific event within the processor, such as the occurrence of an error due to division by zero, which produces an internal interrupt called divide by a zero interrupt. An interrupt in arduino is a signal that tells the processor to immediately stop what it is doing and handle some high priority processing. An interrupt handler is like any other void function. If it is written one and attach it to an interrupt, it will get called whenever that interrupt signal is triggered.

External interruption

There are six external interrupts to serve external devices in arduino mega 2560. Both these interrupts are active low. An external interrupt informs the microcontroller that an external device needs its routine service. External interrupt is a process by which arduino stops its regular task or stop its looping and go to interrupt function to complete its given interrupt function task. They are Digital pin 2, 3, 18, 19, 20, 21 (pins 20 21 are not available to use for interrupts while they are used for I2C communication).

3.10 Conclusion

After presented the competition area, we described briefly the movement of the robot and the techniques used to pick and place the parts in the area.

Chapter 4

Equipement Used

In this chapter, we will present the equipment implemented in the final mechanical design platform, which is based on EMG30 motors and QTR sensor. We end this chapter with the resume of the control process.

4.1 The Controller

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as Pulse Width Modulation (PWM) outputs), 16 analog inputs, 4 Universal Asynchronous Receiver-Transmitter (UART), a 16 MHz crystal oscillator, a Universal Serial Bus (USB) connection, a power jack, an In Circuit Serial Programming (ICSP) header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila. The Mega 2560, shown in Figure 4.1, is an update to the Arduino Mega, which it replaces.

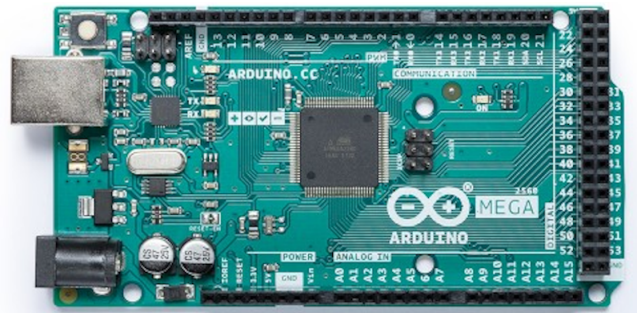

Figure 4.1: Arduino mega

The characteristics of the arduino Mega are shown in Table 4.1.

Microcontroller	ATMEGA2560
Operating Voltage	5 V
Input Voltage	7 V- 12 V
USB Port	Yes
DC Power Jack	Yes
Current Rating Input/Output	20 mA
Current Drawn from Chip	50 mA
Digital Input/Output pins	54
PWM	15
Analog Pins	16
Flash Memory	256 KB
SRAM	8KB
EEPROM	8KB
Crystal Oscillator	16 MHz
LED	Yes
Wi-Fi	No
Shield Compatibility	Yes

Table 4.1: Specification arduino mega 2560

4.1.1 Arduino IDE

The open-source Arduino Software Integrated Development Environment (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board. In this work we install the Arduino IDE into Visual studio code to make

the work easier and clear, Arduino integrated into VS code can do the same things in the Arduino IDE.

4.1.2 Visual Studio Code Envirement

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

First and foremost, it is an editor that gets out of our way. The delightfully frictionless edit-build-debug cycle means less time fiddling with our environment, and more time executing on our ideas.We'll often benefit from tools with more code understanding than just blocks of text. The Arduino IDE is already installed in VS Code in order to more fast compiling and execute the robot tasks.

4.2 EMG30 Motor

The EMG30 gear-motor is a powerful 12V brushed DC motor with a 30: 1 metal gearbox and an integrated quadrature encoder that provides a resolution of 64 counts per revolution of the motor shaft, which corresponds to 1920 counts per revolution of the gearbox's output shaft. These units have a 16 mm-long, 6 mm-diameter D-shaped output shaft. Figure 4.2 shows the EMG30 motor used.

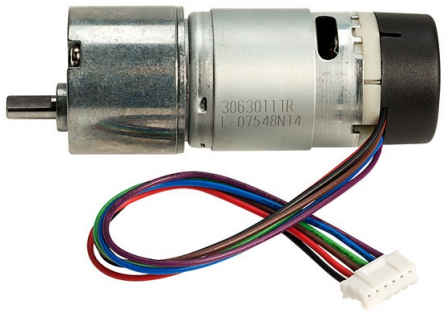

Figure 4.2: EMG30-motor

These motors are intended for use at 12 V, though in general, these kinds of motors

can run at voltages above and below the nominal voltage. Lower voltages might not be practical, and higher voltages could start negatively affecting the life of the motor.

A two-channel Hall effect encoder is used to sense the rotation of a magnetic disk on a rear protrusion of the motor shaft. The quadrature encoder provides a resolution of 64 counts per revolution of the motor shaft when counting both edges of both channels. To compute the counts per revolution of the gearbox output, multiply the gear ratio by 64. The motor encoder has six color-coded, (20 cm) leads terminated by a 1 6 female header with a 0.25 cm pitch, as shown in the main product picture. This header works with standard 0.25 cm male headers and our male jumper and pre-crimped wires. The table 4.2 describes the wire functions.

Red	motor power (connects to one motor terminal)
Black	motor power (connects to the other motor terminal)
Green	GND encoder
Blue	Vcc encoder (3.5 - 20 V)
Yellow	A output encoder
White	B encoder output

Table 4.2: Wire function of EMG30 motor

4.3 The Motor Driver L298N

The L298 H Bridge in the figure 4.3 is base on l298 Chip manufacture by ST Semiconductor. The l298 is an integrated monolithic circuit in a 15 lead multi-watt and power S020 package. It is a high voltage and high current full dual bridge driver designed to accept standard Transistor-Transistor Logic (TTL) logic level and drive inductive loads such as relays, solenoids and DC stepper motor.

Figure 4.3: The motor driver L298N

Two enabled inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together, and the corresponding external terminal can be use for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at lower voltage.

This module has ease to connect and drive a DC motor or stepper motor allows user to easily and intently control two motor up to 2A each in both direction or one stepper motor. It is excellent for robotics applications and well fit to a microcontroller. It can also be interfaced with simple manual switches, TTL Logic gates and relays [12]. The L298N Module Pin Configuration are show in table 4.3.

Pin name	Description
IN1 IN2	Motor A input pins Used to measure the spinning direction of Motor A
IN3 IN4	Motor B input pins Used to measure the spinning direction of Motor B
ENA	Enables PWM signal for Motor A
ENB	Enables PWM signal for Motor B
OUT1 OUT2	Output pins of Motor A
OUT3 OUT4	Output pins of Motor B
12V	12V input from DC power Source
5V	Supplies power for the switching logic circuitry inside L298N IC
GND	Ground pin

Table 4.3: Electro-magnet specifications

4.4 The Electromagnet

An electromagnet is a type of magnet in which the magnetic field is produced by electric current. An electric current flowing in a wire creates a magnetic field around the wire, due to Ampere's law (see drawing below). To concentrate the magnetic field, in an electromagnet the wire is wound into a coil with many turns of wire lying side by side. The magnetic field of all the turns of wire passes through the center of the coil, creating a strong magnetic field there. Grove - Electromagnet in the figure below can shuck 1KG weight and hold on. It is easy to use for pick and place.

Figure 4.4: The Electromagnet

The device must be powered by a voltage source equivalent to 5 V DC, and draws a maximum current of 400 mA. In addition, when the device is in standby mode, it consumes a power of approximately 1 mW.

4.5 QTR-8RC Sensor

This sensor module has 8 IR LED/photo-transistor pairs mounted on a 0.375" pitch, making it a great detector for a line-following robot. Pairs of Light Eitting Diode (LED) are arranged in series to halve current consumption, and a Metal Oxide Semiconductor Field Effect Transistor (MOSFET) allows the LEDs to be turned off for additional sensing or power-savings options. Each sensor provides a separate digital I/O-measurable output.

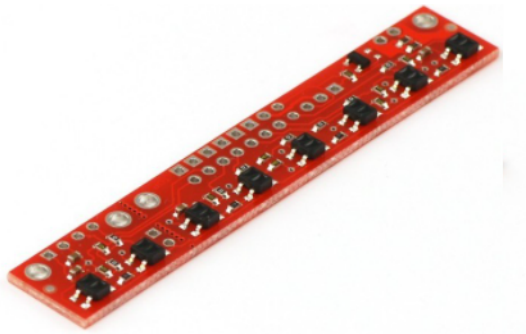

Figure 4.5: QTR-8RC Reflectance Sensor Array

4.6 WIFI Sensor

The ESP8266 is a low-cost Wi-Fi chip with full support for Transmission Control Protocol/Internet Protocol (TCP/IP) and microcontroller functionality. Figure 4.6 shows the ESP8266 Module.

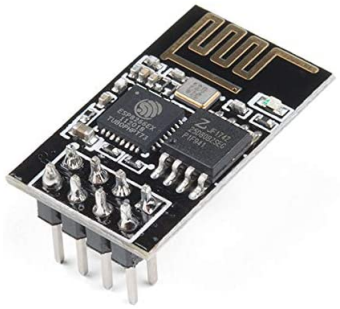

Figure 4.6: Module WIFI

This small module allows microcontrollers to connect to Wi-Fi networks using the TCP/IP protocol through AT commands. However, due to the low cost and the fact that there were very few external components in the module, suggesting that large scale production would be very inexpensive to produce, attracted many enthusiasts to explore the module and the software contained within. The ESP8285 is an ESP8266 with an internal MiB flash memory, which allows single chip devices to connect via Wi-Fi.

4.7 Power Supply

The sizing of the power supply of the robot consists in defining the desired autonomy for normal operation, and secondarily to distribute the capacity on several independent sources. The necessary energy is calculated from the consumption of all the elements of the robot. The calculation is simple, just multiply the total current consumed by the desired autonomy. The power supply can be in the form of batteries or accumulators, taking into account the final weight of the robot, its power and autonomy.

The problem with robots is the lack of space. The choice of battery depends on the mechanical characteristics of the robot.

Figure 4.7: Battery used

Battery specification

Type Battery	**PB Battery**
Capacity	3.2 Ah
Nominal voltage	12 V
Internal resistance	45m
Max Charge Voltage	13.5 to 13.8 V
Standard Charge Current	0.96 A
Weight	1.35 Kg
Operating Temperature	40 degree

Table 4.4: Basic characteristics of the PB battery

4.8 Voltage Step-down

Because the arduino board work on 5 V, and there are no such things as a 3.3 V or 5 V battery, thus, it is needed to include a voltage regulator to step down the voltage offered by the battery and regulate it. Figure 4.8 shows the chosen step down regulator.

This simple power supply board tackles the problem with a buck regulator, the simplest type of switched mode DC/DC converter. It uses a single IC of type LM2596, a flyback diode, and a handful of passive components to set up an efficient voltage regulator. It accepts any DC input voltage between 5V and 30V, making it compatible with just about any battery pack out there. With a screw terminal it's very easy to connect it, and fine tune the output voltage with a trimmer. Most Arduino boards have an integrated voltage regulator, but it's a linear one and thus not very efficient at converting a high voltage to 5V, wasting precious battery power in the process.

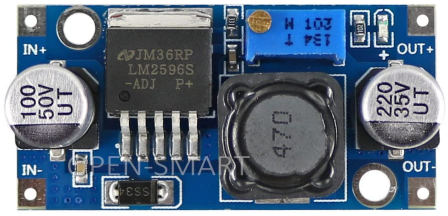

Figure 4.8: LM protection board

4.9 Conclusion

The equipment selection which is a very important part in the elaboration of our project was detailed in this chapter. The work of the two preceding chapters helped for the programming stage of the hardware parts in the following chapter.

Chapter 5

Programming and Realization

The link between the software and hardware components will be introduced in this chapter. Starting with the main task and motion of the robot in the factory lite competition. This chapter will conclude everything to make the robot working in the area requested and ended by perspective program.

5.1 Robot Motion

Forward and back motion

To move the robot forward, the two green 0 wheels must turn clockwise as shown in the figure below. To move backwards, the same two wheels must turn anti-clockwise. This configuration is applied in the arduino code (Odometry part).

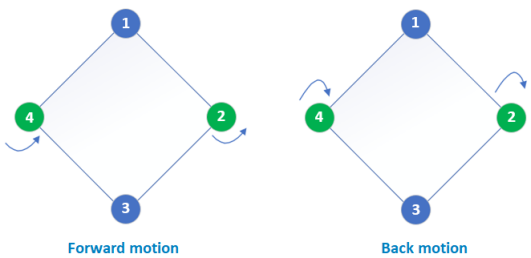

Figure 5.1: Forward and back motion

Left and right motion

To move the robot in the left wise, it need to turn the motor one and three in the anti-clockwise. For the right motion the other two motors should turn in the clockwise.

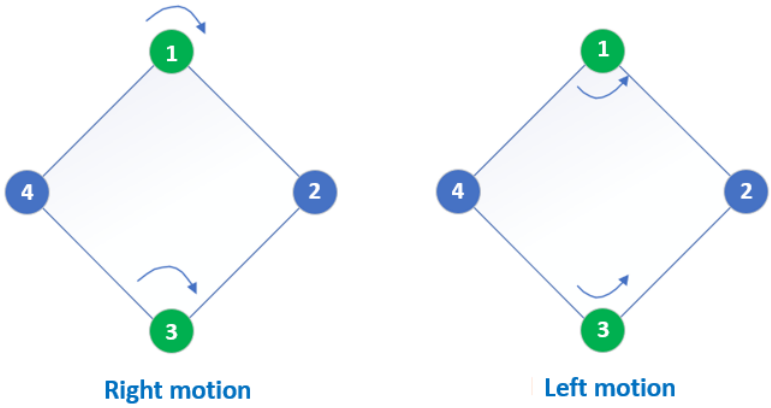

Figure 5.2: Left and right motion

Turn left and right

To turn the robot in the left motion, it need to turn all the motors in the clockwise. For the right motion the other motors should turn in the anti-clockwise as shown in the figure below.

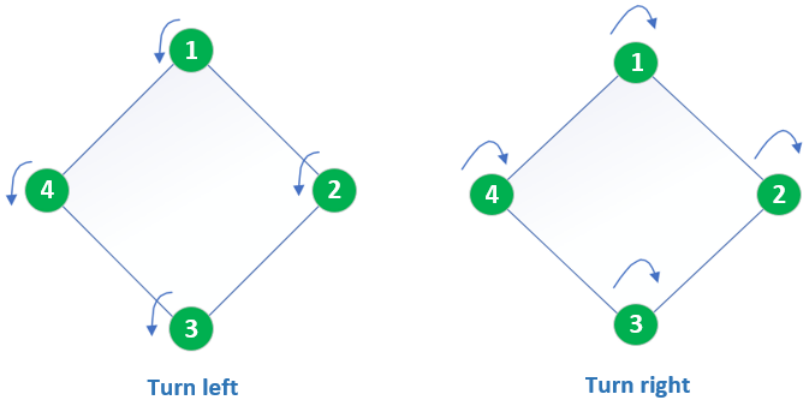

Figure 5.3: Turn left and right

Left and right diagonal

To move the robot diagonally to the left, motors one and two must be turned clockwise and motors three and four anticlockwise. To change the direction we need to reverse the rotation direction of the motors as shown in the figure below.

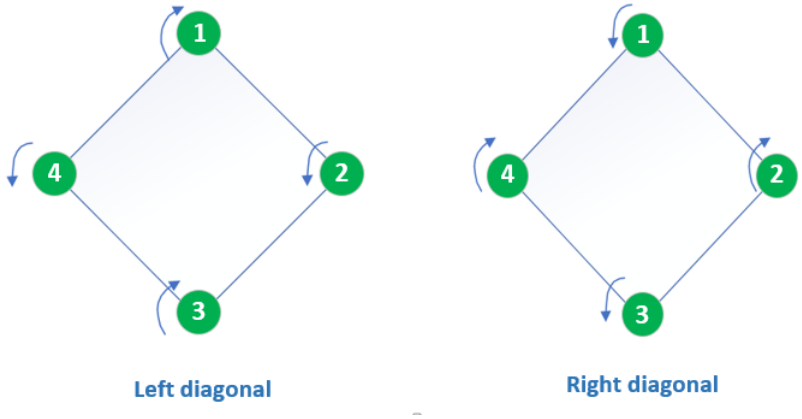

Figure 5.4: Left and right diagonal

5.2 Odemetry

Odometry is the use of data from motion sensors , in this case the sensors are the encoders of the EMG30 motors,To estimate change in position over time. It is used in robotics by some legged or wheeled robots to estimate their position relative to a starting location. Odometry is the most widely used method for determining the momentary position of a mobile robot.

5.3 Odometry Error Model

Lets took as a consideration a mobile robot with a synchronous drive system. Assuming a two-dimensional world, we can define the robot configuration with respect to a world-coordinate frame W by the vector X=[x,y, θ]T, containing its position and orientation. The robot configuration estimated by odometry measurements is different from the actual

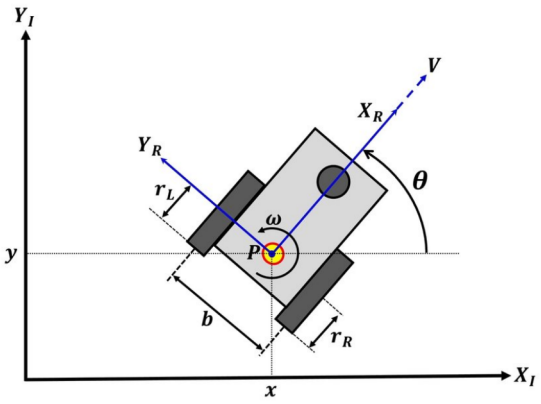

Figure 5.5: Principle of Odometry [13]

configuration X because of the odometry errors.

In order to compute the global odometry error related to a given robot motion, we divided the trajectory in N small segments in figure 5.6. We first modeled the elementary error related to a single segment. Then we computed the cumulative error on the global path. Finally, we took the limit value when N $\rightarrow \infty$.

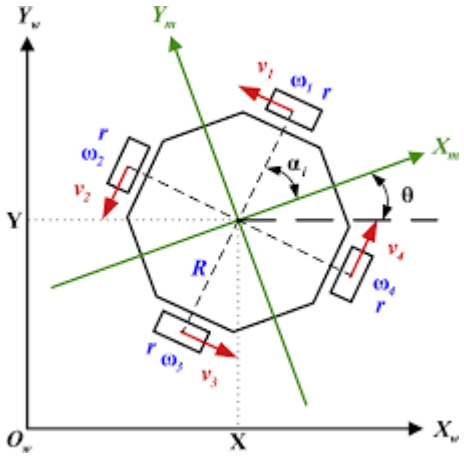

Figure 5.6: Odometry X configuration

5.4 Test Regulation for Odometry

5.4.1 Closed loop control PID

The Proportional Integral Derivative (PID) controller is parameterized in its proportional (Kp), integral (KI) and derivative (KD) gains. Two robust PID design methods based on minimizing the norm of the tracking error caused by a step load disturbance are evaluated to determine the values of these gains [14].

Proportional part

The proportional part reduces the rise time and decreases the steady state error. This means that the system will take lesser time to reach its peak value and when it reaches its steady state, the steady state error will be low. However, it increases the peak overshoot.

Derivative part

The derivative part reduces the overshoot and the settling time. This means that the transient state of the system will be more damped. Also, the system will reach its steady state in a lesser time. However, it does not have any effect on the rise time or the steady state error.

Integral part

The integral part reduces the rise time and completely eliminates the steady state error.

However, it increases the peak overshoot and the settling time.

The figure 5.7 shows the process for coding error correction and PID control for the system.

```c
#include "Robot_Pinout.h"
#include "Robot_Conf.h"
#include "Robot_Regulation.h"


void RobotRegulation_Init(RobotRegulation_Handle *hRegulation)
{
    memset(hRegulation, 0, sizeof(*hRegulation));
    hRegulation->Kp = KP_MOVE;
    hRegulation->Ki = KI_MOVE;
    hRegulation->Kd = KD_MOVE;
}

int RobotRegulation_Compute(RobotRegulation_Handle *hRegulation, int DesiredPosition, int CurrentPosition)
{
  int Output = 0;

  hRegulation->CurrentTime  = millis() ;
  hRegulation->DeltaTime    = hRegulation->CurrentTime - hRegulation->previousTime ;
  hRegulation->Error        = DesiredPosition - CurrentPosition ;
  hRegulation->Ierror      += hRegulation->Error * hRegulation->DeltaTime ;
  hRegulation->Derror       = (hRegulation->Error - hRegulation->LastError)/hRegulation->DeltaTime ;
  hRegulation->Perror       = hRegulation->Error * hRegulation->Kp ;
  Output                    = hRegulation->Perror + hRegulation->Ierror * hRegulation->Ki - hRegulation->Derror *hRegulation->Kd ;
  Output                    = abs(Output);
  hRegulation->LastError    = hRegulation->Error ;
  hRegulation->previousTime = hRegulation->CurrentTime ;

  if(Output < ROBOT_MIN_VELOCITY)Output    = ROBOT_MIN_VELOCITY ;
  if(Output > ROBOT_MAX_VELOCITY) Output   = ROBOT_MAX_VELOCITY ;

  return Output;
}
```

Figure 5.7: Robot regulation part

5.4.2 Find Distance with Encoders

As each motor shaft rotates, it also rotates its attached ring magnet at the same rate. As the ring magnet completes one full rotation, the Hall effect sensor detects 4 changes (or "ticks") in the magnetic field as each magnetic pole passes by the sensor. However, each rotation of the motor only turns the wheel a certain number of degrees. The EMG30 motors have a gearbox ratio of 30:1, which means it takes 30 rotations of the motor to turn the wheel one complete revolution (360 °).

for one shaft of the motor , the wheel move distance equal to 180 mm ,in the figure 5.8 checking the rotation sens and value of the encoder measurement. Based on the size

of the robot's wheels, we can also calculate the distance that the robot travels .

```
void blink()
{
if ( digitalRead (ENCODER_A_MOTOR1)==digitalRead (ENCODER_B_MOTOR1))
TICK1++;
else
TICK1--;
Serial.println(TICK1);
}

void blink2()
{

if ( digitalRead (ENCODER_A_MOTOR2)==digitalRead (ENCODER_B_MOTOR2))
TICK3++;
else
TICK3--;
Serial.println(TICK3);
}
void blink3()
{

if ( digitalRead (ENCODER_A_MOTOR3)==digitalRead (ENCODER_B_MOTOR3))
TICK4++;
else
TICK4--;
Serial.println(TICK4);
}
```

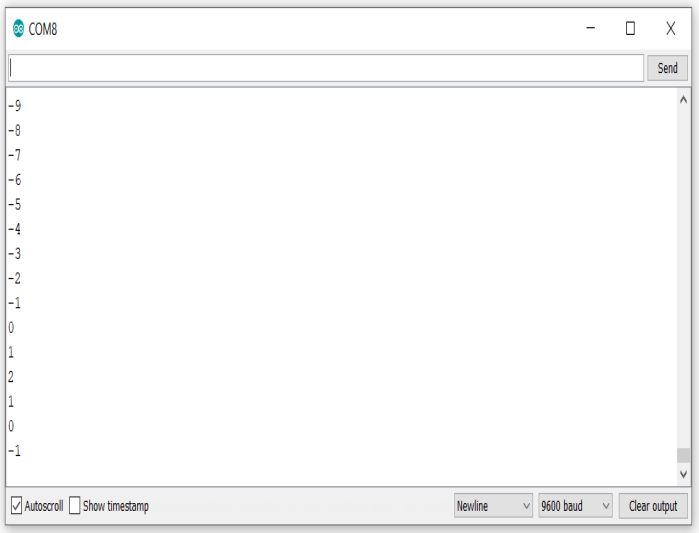

Figure 5.8: Encoder test sens

5.4.3 Check Encoder Positions

In order to function accurately, each wheel encoder sensor must be positioned correctly, relative to its ring magnet. The sensor tip must be centered within the silver band of the ring magnet (not too far inward or outward) and must be close to the ring magnet's surface (about 1/8 inch away). Visually check the position of the left and right encoder sensors. If necessary, we might need to push (or pull) a sensor to position it correctly. Each encoder need to be initialized or get the actual value of the position and send it to the arduino as indicated in the figure 5.9.

Robot_Encoder.h

main > Robot_Encoder.h > __unnamed_enum_0002_1

```c
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __ROBOT_ENCODER_H
#define __ROBOT_ENCODER_H

#ifdef __cplusplus
extern "C" {
#endif


typedef enum
{
    ENCODER_MOTOR1,
    ENCODER_MOTOR2,
    ENCODER_MOTOR3,
    ENCODER_MOTOR4,
}Encoder_TypeDef;

void RobotEncoders_Init(void);
int RobotEncoders_SetValue(Encoder_TypeDef Encoder, int Value);
int RobotEncoders_GetValue(Encoder_TypeDef Encoder);
#ifdef __cplusplus
}
#endif

#endif /* __ROBOT_CONF_H */
```

Figure 5.9: Encoder robot function

5.4.4 Encoders Programming

To use the wheel encoders of the EMG30 motor on thefour omnidirectional mobile robot, it is necessary to: Change the number of ticks to distance and angle (it depends of the mechanical dimension of the components of the robot).Add code statement to drive one or both motors Use the object's getValue() method to get the current encoder counts. Add a sequence statement to perform action based on the encoder counts.

5.4.5 Measuring Movement

The technique of measuring the movement of the position of this robot, called odometry, requires an encoder that translates the turn of the wheels into the corresponding traveled distance. To provide four degrees of freedom while moving, its typically need four motors, and as described in the figure 5.10.

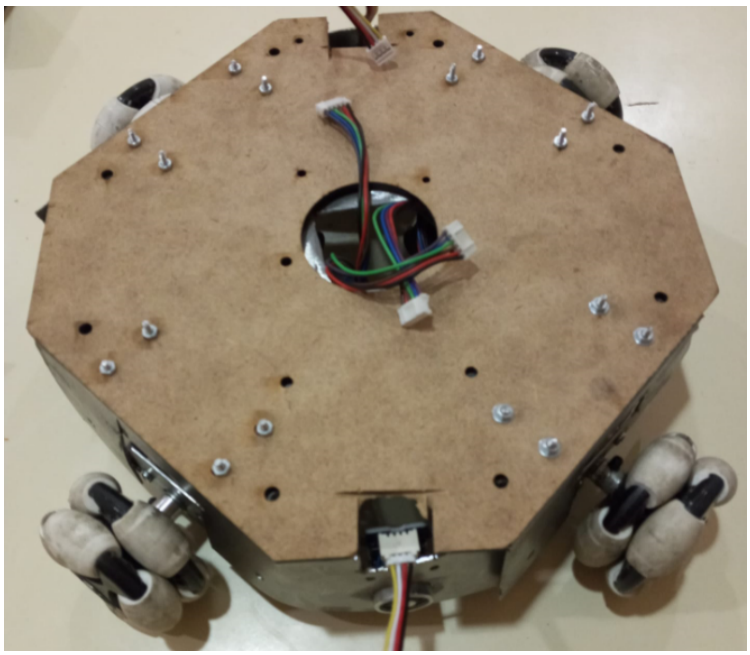

Figure 5.10: Mechanical structure

The equations for computing the position from the decoded movements depends on the architecture of the robot are explained in chapter 2. We will explain it here using the example of the differential drive. Referring to the robot structure dimension and the dimension of the omni wheels, it is possible to define the distance desired. The next figure show how does the robot calculate and change the number of ticks to the desired position.

```
#define DISTANCE_TO_TICK(Distance)     (unsigned long)(((unsigned long)Distance)*180)/188
#define ANGLE_TO_TICK(Angle)           (unsigned long)(((unsigned long)Angle)*81)/45
typedef struct
{
    void(*Action)(int , int );
    int Motor;
    unsigned long DesiredPosition;
    int WaitTimeMs;
}RobotSequence_Typedef;

void RobotSequenceInit(RobotSequence_Typedef* pRobotSequence);
void RobotSequenceExecuteNextSeq(void);
#ifdef __cplusplus
}
#endif

#endif /* __ROBOT_SEQUENCE_H */
```

Figure 5.11: Measure movement

5.5 Following The Line Using The QTR Sensor

5.5.1 Line detection

It was known that radiation of a certain wavelength will be absorbed by a material, if the material diffuses all radiation except that of this wavelength, and it will be diffused by an other material diffusing radiation.In our case, it was advisable to use two "colors": black which absorbs all wavelengths of radiation, and the white, which diffuses all wavelengths, including infrared. As for the QTR sensor implemented in the robot, it will only have to be able to detect the absence or the presence of the black line . It is necessary to take in consideration the environment of the robot, which is not necessarily in total darkness; the only significant sources of light in a room of the competition come from neon lights or lamps. with incandescence whose emitted infra-red does not disturb the robot. The QTR

sensors that was used will be associated with ultrasnic sensor that emit the presence of the parts in the area of the competition. The magnet installed between each two wheels will be the actuator to pick and place the parts in the right position.

5.5.2 QTR code test

To follow the line with the stable motion of the robot, it is important to adjust the right KP value for the PID controller of the robot as indicated in the figure below.The right value found with sampling in many real test of the robot motion on the black line and calibration of the QTR sensor.

```
void setup()
{
  pinMode (MOTOR1_IN1, OUTPUT);
  pinMode (MOTOR1_IN2, OUTPUT);

  pinMode (MOTOR2_IN1, OUTPUT);
  pinMode (MOTOR2_IN2, OUTPUT);

   pinMode (MOTOR1_EN, OUTPUT);
  pinMode (MOTOR2_EN, OUTPUT);

  // configure the sensors
  qtr.setTypeAnalog();
  qtr.setSensorPins((const uint8_t[]){A0, A1, A2, A3, A4, A5,A6,A7}, SensorCount);
  qtr.setEmitterPin(2);

  digitalWrite(LED_BUILTIN, LOW); // turn off Arduino's LED to indicate we are through with calibra

  // print the calibration minimum values measured when emitters were on
  Serial.begin(9600);
}
int setpoint = 3000;
float Kp =0.1,Ki =0.0, Kd = 0   ,Derror, Ierror,LastError ; // motors Kp =1.0,Ki =0.001, Kd = 1

int CurrentTime ,DeltaTime,previousTime ;
int Error=0,Perror = 0,Output,M1,M2;

uint16_t positions ;
```

Figure 5.12: QTR set up program

```
void loop()
{
    qtr.read(sensorValues);

 positions = qtr.readLineBlack(sensorValues);
  CurrentTime = millis() ;
  DeltaTime = CurrentTime - previousTime ;
  Error = setpoint - positions ;

  Ierror  += Error * DeltaTime ;
  Derror = (Error - LastError)/DeltaTime ;
  Perror = Error * Kp ;

  Output = Perror + Ierror * Ki - Derror *Kd ;

 M1 = 200 + Output ;
   M2 = 200 - Output ;
  if(M1 < 100)M1 = 100 ;
   if(M1 > 255)M1 = 255 ;
   if(M2 < 100)M2 = 100 ;
   if(M2 > 255)M2 = 255 ;
Serial.println(M1);
      //Motor 1 Avance
     analogWrite(MOTOR1_EN, M1);
     digitalWrite(MOTOR1_IN1, HIGH);
     digitalWrite(MOTOR1_IN2, LOW);
      analogWrite(MOTOR2_EN, M2);
     digitalWrite(MOTOR2_IN1, LOW);
     digitalWrite(MOTOR2_IN2, HIGH);
  LastError = Error ;
  previousTime = CurrentTime ;
//  Serial.println(Error);
  delay(5);
```

Figure 5.13: Loop of QTR sensor

5.6 Conclusion

The programming stage has been well thought out and carried out ,We have tried to get as close as possible to the operating mode.This chapter contains the new program and a general algorithm for control and during the robotic competition.

General Conclusion and Future Work

General Conclusion

Mobile robots are used for automatically transporting products in factories and warehouses. Especially, omnidirectional mobile robots that can move immediately in an arbitrary direction have the potential for further efficiency. However, existing omnidirectional mobile robots need specialized wheel mechanisms, which can be unreliable. To solve this problem. The idea of the robot which has been presented in this project employs instructions from sensors and on board arduino to achieve its physical movement. One of its significant attribute is controlling efficiently with very much accuracy. It does not use complex algorithms for line following applications , even for pick and place the materials exist in the area of the competition. Controlling process has been made automatic by straightforward controlling mechanism. Simple basic electronics is used instead of costly microcontrollers which made it very much cost effective. Further modification of this robot includes additional sensors like WI-FI and infrared so that the robot will be able to follow a line or use odometry having the ability to pick and place the parts exist.

In its current form robot is enough capable. It can follow any curve and cycle line. We must build a robot that has light weigh and high speed because points are awarded based upon the distance covered and the speed of the overall robot. Therefore, we used four EMG30 motors.

The body weight and wheels radius have effects on speed, too. The weight of the designed robot is around 4kg and it can be lighter. To get better maneuver, we must build a robot that uses two motors and two wheels on the rear and a free wheel on the front. The power supply is 12 V with regulator.

The designed robot has QTR sensors on the bottom for detect line. Microcontroller Arduino Mega 2560 R2 and drivers L298 were used to control direction and speed of the four motors.

Future Work

The future work will be to attach a manipulator arm to the omnidirectional wheel mobile robot, the main things to do is to make communication between the base and the brake arm.

The objective is to develop more and more about the mobility and activities of the mobile robot attached to the arm. On the surface of the robot structure, a place is prepared to install this arm above the robot.

Our futur design and assembling robot is intended for research and education , it is similar to the KUKA Youbot mobile robot. KUKA well known as one of the world's leading industrial robots manufacturer has designed a rich platform, affordable and open.With a small KUKA arm with 5 degrees of freedom.

Figure 5.14: Braccio arduino arm

Bibliography

[1] Lab4sys. (consulted on 22 october 2020). 'History of robotucs', [Online]. Available: https://lab4sys.com/en/history-of-robotics/?cn-reloaded=1.

[2] G. A. Demetriou, "Mobile robotics in education and research", *Mobile Robots-Current Trends*, pp. 27–48, 2011.

[3] Arduino. (consulted on 22 march 2021). 'attachInterrupt()', [Online]. Available: https://roboticsandautomationnews.com/2017/09/14/kuka-launches-autonomous-mobile-robot-for-logistics-industry/14077/.

[4] MHI. (consulted on 25 march 2020). 'Industrial robots, [Online]. Available: https://www.mhi.org/fundamentals/robots.

[5] L. Lynch, T. Newe, J. Clifford, J. Coleman, J. Walsh, and D. Toal, "Automated Ground Vehicle (AGV) and Sensor Technologies-A Review", in *2018 12th International Conference on Sensing Technology (ICST)*, IEEE, 2018, pp. 347–352.

[6] D. Kragic, J. Gustafson, H. Karaoguz, P. Jensfelt, and R. Krug, "Interactive, Collaborative Robots: Challenges and Opportunities.", in *IJCAI*, 2018, pp. 18–25.

[7] Y. Liu, X. Wu, J. J. Zhu, and J. Lew, "Omni-directional mobile robot controller design by trajectory linearization", in *Proceedings of the 2003 American Control Conference, 2003.*, IEEE, vol. 4, 2003, pp. 3423–3428.

[8] B.-S. Cho, W.-J. Seo, W.-s. Moon, and K.-R. Baek, "Positioning of a mobile robot based on odometry and a new ultrasonic LPS", *International Journal of Control, Automation and Systems*, vol. 11, no. 2, pp. 333–345, 2013.

[9] J. Gonçalves, J. Lima, P. Costa, and A. Moreira, "Manufacturing Education and Training resorting to a new mobile robot competition", in *Conference on Flexible Automation and Intelligent Manufacturing FAIM 2012*, Tampere University of Technology, 2012.

[10] R. factory lite. (consulted on 2019). robot factory lite, [Online]. Available: `https://web.fe.up.pt/~robotica2019/index.php/pt/robot-factory-lite`.

[11] Arduino. (consulted on 22 march 2021). 'attachInterrupt()', [Online]. Available: `https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/`.

[12] 1. core. (consulted on 2018). using l298N half bridge, [Online]. Available: `https://www.14core.com/wiring-driving-the-l298n-h-bridge-with-stepper-motors/`.

[13] N. Chindakham, Y.-Y. Kim, A. Pirayawaraporn, and M.-H. Jeong, "Simultaneous Calibration of Odometry and Head-Eye Parameters for Mobile Robots with a Pan-Tilt Camera", *Sensors*, vol. 19, no. 16, p. 3623, 2019.

[14] K. Soltesz, J.-O. Hahn, T. Hägglund, G. A. Dumont, and J. M. Ansermino, "Individualized closed-loop control of propofol anesthesia: A preliminary study", *Biomedical Signal Processing and Control*, vol. 8, no. 6, pp. 500–508, 2013.

Appendix A

Arduino Code

```
#ifndef __ROBOT_REGULATION_H
#define __ROBOT_REGULATION_H
#ifdef __cplusplus
extern "C" {
#endif
typedef struct {
   float Kp;
   float Ki;
   float Kd;
   float Derror;
   float Ierror;
   float LastError;
   int Error;
   int Perror;
   int CurrentTime;
   int DeltaTime;
   int previousTime;
}RobotRegulation_Handle;
void RobotRegulation_Init(RobotRegulation_Handle *hRegulation);
int  RobotRegulation_Compute(RobotRegulation_Handle *hRegulation, int DesiredPosition, int
CurrentPosition);
#ifdef __cplusplus    }
#endif
#endif
```

Robot Regulation

```
#include "Robot_Pinout.h"

#include "Robot_Conf.h"

#include "Robot_Regulation.h"

void RobotRegulation_Init(RobotRegulation_Handle *hRegulation)

{

  memset(hRegulation, 0, sizeof(*hRegulation));

  hRegulation->Kp = KP_MOVE;

  hRegulation->Ki = KI_MOVE;

  hRegulation->Kd = KD_MOVE;

}

int RobotRegulation_Compute(RobotRegulation_Handle *hRegulation, int DesiredPosition, int Curre
ntPosition)

{

 int Output = 0;

 hRegulation->CurrentTime  = millis() ;

 hRegulation->DeltaTime    = hRegulation->CurrentTime - hRegulation->previousTime ;

 hRegulation->Error     = DesiredPosition - CurrentPosition ;

 hRegulation->Ierror    + = hRegulation->Error * hRegulation->DeltaTime ;

 hRegulation->Derror   = (hRegulation->Error - hRegulation->LastError)/hRegulation->DeltaTime ;

 hRegulation->Perror   = hRegulation->Error * hRegulation->Kp ;

 Output    = hRegulation->Perror + hRegulation->Ierror * hRegulation->Ki - hRegulation-
>Derror *hRegulation->Kd ;

 Output    = abs(Output);

 hRegulation->LastError    = hRegulation->Error ;

 hRegulation->previousTime = hRegulation->CurrentTime ;

 if(Output < ROBOT_MIN_VELOCITY)Output    = ROBOT_MIN_VELOCITY ;

 if(Output > ROBOT_MAX_VELOCITY) Output   = ROBOT_MAX_VELOCITY ;

  return Output;  }
```

Robot Actions

```
#if (ARDUINO >= 100)
#include <Arduino.h>
#else
#include <WProgram.h>
#endif
#include "Robot_Actions.h"
#include "Robot_Motors.h"
#include "Robot_Encoder.h"
#include "Robot_ElectroMagnet.h"
#include "Robot_Conf.h"
static RobotActionHandle_TypeDef hRobot ;
static void RobotActionMotor1(int DesiredPosition);
static void RobotActionMotor2(int DesiredPosition);
static void RobotActionMotor3(int DesiredPosition);
static void RobotActionMotor4(int DesiredPosition);
void RobotActionInit(void)       {
   RobotRegulation_Init(&hRobot.hRegulation1);
   RobotRegulation_Init(&hRobot.hRegulation2);
   RobotRegulation_Init(&hRobot.hRegulation3);
   RobotRegulation_Init(&hRobot.hRegulation4);
   RobotEncoders_Init();
   RobotMotorInit();
   RobotElectromagnetInit();    }
void RobotActionMove(int Motor, int DesiredPosition)
{     /*Check if motor 1 is selected*/
  if((Motor & ROBOT_ACTION_MOTOR_1) == ROBOT_ACTION_MOTOR_1)
  { RobotEncoders_SetValue(ENCODER_MOTOR1, 0);
    hRobot.Motor1State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor1DesiredPosition = DesiredPosition;       }
```

```
 /*Check if motor 2 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_2) == ROBOT_ACTION_MOTOR_2)
 {     RobotEncoders_SetValue(ENCODER_MOTOR2, 0);
   hRobot.Motor2State = MOTOR_ACTION_STATE_IN_PROGRESS;
   hRobot.Motor2DesiredPosition = DesiredPosition;    }
 /*Check if motor 3 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_3) == ROBOT_ACTION_MOTOR_3)
 {    RobotEncoders_SetValue(ENCODER_MOTOR3, 0);
   hRobot.Motor3State = MOTOR_ACTION_STATE_IN_PROGRESS;
   hRobot.Motor3DesiredPosition = DesiredPosition;     }
 /*Check if motor 4 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_4) == ROBOT_ACTION_MOTOR_4)
 {    RobotEncoders_SetValue(ENCODER_MOTOR4, 0);
   hRobot.Motor4State = MOTOR_ACTION_STATE_IN_PROGRESS;
   hRobot.Motor4DesiredPosition = DesiredPosition;     }     }
void RobotActionMoveDiagonal(int Motor, int DesiredPosition)
{      /*Check if motor 1 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_1) == ROBOT_ACTION_MOTOR_1)     {
   RobotEncoders_SetValue(ENCODER_MOTOR1, DesiredPosition);
   hRobot.Motor1State = MOTOR_ACTION_STATE_IN_PROGRESS;
   hRobot.Motor1DesiredPosition = 0;    }
 /*Check if motor 2 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_2) == ROBOT_ACTION_MOTOR_2)
 {    RobotEncoders_SetValue(ENCODER_MOTOR2, 0);
   hRobot.Motor2State = MOTOR_ACTION_STATE_IN_PROGRESS;
   hRobot.Motor2DesiredPosition = DesiredPosition;    }
 /*Check if motor 3 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_3) == ROBOT_ACTION_MOTOR_3)   {
   RobotEncoders_SetValue(ENCODER_MOTOR3, 0);
```

```
    hRobot.Motor3State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor3DesiredPosition = DesiredPosition;    }
  /*Check if motor 4 is selected*/
  if((Motor & ROBOT_ACTION_MOTOR_4) == ROBOT_ACTION_MOTOR_4)
  { RobotEncoders_SetValue(ENCODER_MOTOR4, DesiredPosition);
    hRobot.Motor4State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor4DesiredPosition = 0;  } }
void RobotActionBack(int Motor, int DesiredPosition)
{   /*Check if motor 1 is selected*/
  if((Motor & ROBOT_ACTION_MOTOR_1) == ROBOT_ACTION_MOTOR_1)   {
    RobotEncoders_SetValue(ENCODER_MOTOR1, DesiredPosition);
    hRobot.Motor1State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor1DesiredPosition = 0;  }
  /*Check if motor 2 is selected*/
  if((Motor & ROBOT_ACTION_MOTOR_2) == ROBOT_ACTION_MOTOR_2)
  {  RobotEncoders_SetValue(ENCODER_MOTOR2, DesiredPosition);
    hRobot.Motor2State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor2DesiredPosition = 0;    }
  /*Check if motor 3 is selected*/
  if((Motor & ROBOT_ACTION_MOTOR_3) == ROBOT_ACTION_MOTOR_3)
  {RobotEncoders_SetValue(ENCODER_MOTOR3, DesiredPosition);
    hRobot.Motor3State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor3DesiredPosition = 0;    }
  /*Check if motor 4 is selected*/
  if((Motor & ROBOT_ACTION_MOTOR_4) == ROBOT_ACTION_MOTOR_4)
  {   RobotEncoders_SetValue(ENCODER_MOTOR4, DesiredPosition);
    hRobot.Motor4State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor4DesiredPosition = 0;   }   }
void RobotActionTurnRight(int Motor, int DesiredPosition)   {
  /*Check if motor 1 is selected*/
```

```
 if((Motor & ROBOT_ACTION_MOTOR_1) == ROBOT_ACTION_MOTOR_1)
 {   RobotEncoders_SetValue(ENCODER_MOTOR1, 0);
   hRobot.Motor1State = MOTOR_ACTION_STATE_IN_PROGRESS;
   hRobot.Motor1DesiredPosition = DesiredPosition;    }
 /*Check if motor 2 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_2) == ROBOT_ACTION_MOTOR_2)
 {   RobotEncoders_SetValue(ENCODER_MOTOR2, 0);
    hRobot.Motor2State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor2DesiredPosition = DesiredPosition;   }
 /*Check if motor 3 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_3) == ROBOT_ACTION_MOTOR_3)
 {    RobotEncoders_SetValue(ENCODER_MOTOR3, DesiredPosition);
    hRobot.Motor3State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor3DesiredPosition = 0;    }
 /*Check if motor 4 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_4) == ROBOT_ACTION_MOTOR_4)
 {   RobotEncoders_SetValue(ENCODER_MOTOR4, DesiredPosition);
    hRobot.Motor4State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor4DesiredPosition = 0;  }   }
void RobotActionTurnLeft(int Motor, int DesiredPosition)   {
 /*Check if motor 1 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_1) == ROBOT_ACTION_MOTOR_1)
 {   RobotEncoders_SetValue(ENCODER_MOTOR1, DesiredPosition);
   hRobot.Motor1State = MOTOR_ACTION_STATE_IN_PROGRESS;
   hRobot.Motor1DesiredPosition = 0;  }
 /*Check if motor 2 is selected*/
 if((Motor & ROBOT_ACTION_MOTOR_2) == ROBOT_ACTION_MOTOR_2)
 {    RobotEncoders_SetValue(ENCODER_MOTOR2, DesiredPosition);
    hRobot.Motor2State = MOTOR_ACTION_STATE_IN_PROGRESS;
    hRobot.Motor2DesiredPosition = 0;   }
```

```
  /*Check if motor 3 is selected*/
  if((Motor & ROBOT_ACTION_MOTOR_3) == ROBOT_ACTION_MOTOR_3)
  {   RobotEncoders_SetValue(ENCODER_MOTOR3, 0);
     hRobot.Motor3State = MOTOR_ACTION_STATE_IN_PROGRESS;
     hRobot.Motor3DesiredPosition = DesiredPosition;     }
  /*Check if motor 4 is selected*/
  if((Motor & ROBOT_ACTION_MOTOR_4) == ROBOT_ACTION_MOTOR_4)
  {  RobotEncoders_SetValue(ENCODER_MOTOR4, 0);
     hRobot.Motor4State = MOTOR_ACTION_STATE_IN_PROGRESS;
     hRobot.Motor4DesiredPosition = DesiredPosition;   }   }
void RobotActionEnableElectromagnet(int Electromagnet, int dummy)
{   /*Avoid warning*/
  (void)dummy;
  if((Electromagnet & ROBOT_ACTION_ELECTRO_MAGNET_1) ==
ROBOT_ACTION_ELECTRO_MAGNET_1)     {
    RobotElectroMagnet1On();  }
  if((Electromagnet & ROBOT_ACTION_ELECTRO_MAGNET_2) ==
ROBOT_ACTION_ELECTRO_MAGNET_2)    {
    RobotElectroMagnet2On();    }   }
void RobotActionDisableElectromagnet(int Electromagnet, int dummy)  {
  /*Avoid warning*/
  (void)dummy;
  if((Electromagnet & ROBOT_ACTION_ELECTRO_MAGNET_1) ==
ROBOT_ACTION_ELECTRO_MAGNET_1)   {
    RobotElectroMagnet1Off();   }
  if((Electromagnet & ROBOT_ACTION_ELECTRO_MAGNET_2) ==
ROBOT_ACTION_ELECTRO_MAGNET_2)  {
    RobotElectroMagnet2Off();  } }
MotorAction_TypeDef RobotActionCompute(void) {
  /*Check if motor 1 is selected*/
  if(hRobot.Motor1State != MOTOR_ACTION_STATE_IDLE)  {
```

```
   RobotActionMotor1(hRobot.Motor1DesiredPosition);

   /*check if Robot doesn't move anymore*/

   if(abs((hRobot.Motor1DesiredPosition -  RobotEncoders_GetValue(ENCODER_MOTOR1))) <=
ROBOT_OFFSET)   {

      hRobot.Motor1State = MOTOR_ACTION_STATE_COMPLETE;  }   }

  /*Check if motor 2 is selected*/

  if(hRobot.Motor2State != MOTOR_ACTION_STATE_IDLE)   {

    RobotActionMotor2(hRobot.Motor2DesiredPosition);

    /*check if Robot doesn't move anymore*/

   if(abs((hRobot.Motor2DesiredPosition -  RobotEncoders_GetValue(ENCODER_MOTOR2))) <=
ROBOT_OFFSET)   {

      hRobot.Motor2State = MOTOR_ACTION_STATE_COMPLETE;   } }

  /*Check if motor 3 is selected*/

  if(hRobot.Motor3State != MOTOR_ACTION_STATE_IDLE)  {

    RobotActionMotor3(hRobot.Motor3DesiredPosition);

    /*check if Robot doesn't move anymore*/

   if(abs((hRobot.Motor3DesiredPosition -  RobotEncoders_GetValue(ENCODER_MOTOR3))) <=
ROBOT_OFFSET)   {

      hRobot.Motor3State = MOTOR_ACTION_STATE_COMPLETE;  }   }

  /*Check if motor 4 is selected*/

  if(hRobot.Motor4State != MOTOR_ACTION_STATE_IDLE)  {

    RobotActionMotor4(hRobot.Motor4DesiredPosition);

    /*check if Robot doesn't move anymore*/

   if(abs((hRobot.Motor4DesiredPosition -  RobotEncoders_GetValue(ENCODER_MOTOR4))) <=
ROBOT_OFFSET)   {

      hRobot.Motor4State = MOTOR_ACTION_STATE_COMPLETE;   }   }

  delay(REGULARION_STEPS);

  if((hRobot.Motor1State == MOTOR_ACTION_STATE_IN_PROGRESS) ||\

    (hRobot.Motor2State == MOTOR_ACTION_STATE_IN_PROGRESS) ||\

    (hRobot.Motor3State == MOTOR_ACTION_STATE_IN_PROGRESS) ||\

    (hRobot.Motor4State == MOTOR_ACTION_STATE_IN_PROGRESS) )    {
```

```
    return MOTOR_ACTION_ONGOING;   }
  return MOTOR_ACTION_DONE;  }
static void RobotActionMotor1(int DesiredPosition)  {
  hRobot.Motor1SpeedToSet = RobotRegulation_Compute(&hRobot.hRegulation1,DesiredPosition,
                                    RobotEncoders_GetValue(ENCODER_MOTOR1));
    if(hRobot.hRegulation1.Error > 0)  {
      RobotMotor1Move(hRobot.Motor1SpeedToSet);   }
    Else   {
      RobotMotor1Back(hRobot.Motor1SpeedToSet);   }  }
static void RobotActionMotor2(int DesiredPosition)  {
  hRobot.Motor2SpeedToSet = RobotRegulation_Compute(&hRobot.hRegulation2,DesiredPosition,
                                    RobotEncoders_GetValue(ENCODER_MOTOR2));
    if(hRobot.hRegulation2.Error > 0)  {
      RobotMotor2Move(hRobot.Motor2SpeedToSet);  }
    Else  {
      RobotMotor2Back(hRobot.Motor2SpeedToSet);  }  }
static void RobotActionMotor3(int DesiredPosition)  {
  hRobot.Motor3SpeedToSet = RobotRegulation_Compute(&hRobot.hRegulation3,DesiredPosition,
                                    RobotEncoders_GetValue(ENCODER_MOTOR3));
    if(hRobot.hRegulation3.Error > 0) {
      RobotMotor3Move(hRobot.Motor3SpeedToSet);  }
    Else  {
      RobotMotor3Back(hRobot.Motor3SpeedToSet);  }  }
static void RobotActionMotor4(int DesiredPosition) {
  hRobot.Motor4SpeedToSet = RobotRegulation_Compute(&hRobot.hRegulation4, DesiredPosition,
                                    RobotEncoders_GetValue(ENCODER_MOTOR4));
    if(hRobot.hRegulation4.Error > 0)  {
      RobotMotor4Move(hRobot.Motor4SpeedToSet); }
    Else {
      RobotMotor4Back(hRobot.Motor4SpeedToSet);   }  }
```